Using the Unreal game engine for Multi-User-Simulation – First impressions

Kilian Gröne[1], Michaela Rehm[1], Donaji Martinez Garcia[1], Gerald Temme[1], Laura Quante[1], Marvin Achilles[1], Martin Fischer[1]

(1) German Aerospace Center - DLR, Lilienthalplatz 7, 38108 Braunschweig, Germany, e-mail: {kilian.groene, michaela.rehm, donaji.martinezgarcia, gerald.temme, laura.quante, ma.fischer, marvin.achilles}@dlr.de

Abstract *– Multi-User-Simulation is a tool for studying social interactions in traffic situations, including vehicles and vulnerable road users. The presented simulators use Unreal Engine as visualization software. The two methods used are a message-broker approach and the integrated multiplayer framework of the Unreal Engine. These approaches were carried out in the MoSAIC-VRU laboratory of the DLR - Institute of Transportation Systems.*

Keywords: *Multi-User-Simulation, Human-centred Simulation, Unreal Game Engine, Vulnerable Road Users*

Introduction

Simulation-based research is critical in transportation science because it provides a safe and controlled environment for analyzing roadway hazards. Human-in-the-loop simulation focuses on studying the behavior of a single participant, allowing evaluation of road infrastructure and assistance systems. However, when considering interactions with other road users, such as merging or intersections, there is a need for real road user behavior. This requires the combination of multiple human-in-the-loop simulators in a multi-user simulation, including vulnerable road users such as cyclists and pedestrians. This allows the analysis of social interactions that play an important role in road traffic. (Muehlbacher, 2015)

Another enhancement to simulation-based research is the use of state-of-the-art game engines. (Chance, et al., 2022) In addition to the relatively accessible, high-quality graphics of these engines and affordable access to virtual reality (VR) and many other features, some game engines offer a multiplayer/networking framework that enables distributed application. This work aims to show the advantages of using game engines at the example of the Unreal game engine.

Simulator Setups

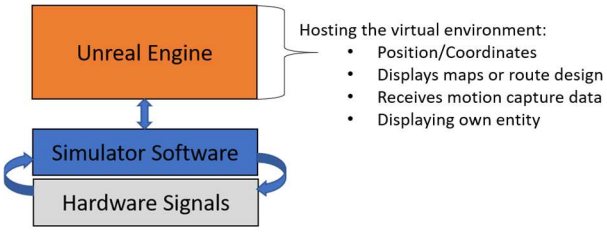

Figure 1 General Setup

The general structure of the simulators used, is shown in Figure 1.

The hardware control is closely linked to the simulation software to minimize latencies. The visualization software (Unreal Engine 5) runs separately on dedicated hardware and communicates with the core software via UDP messages. The decision of whether to implement certain functions in Unreal Engine or the simulation software depended on factors such as the physical behavior of the means of transport and the safety requirements.

For the bicycle simulator, the physical model was developed outside Unreal Engine to meet hardware requirements and ensure safety and reliability. The simulator software sends speed, rotation, steering angle, and leaning angle data to the visualization, while position changes in the virtual environment are primarily determined by speed and yaw rate.

On the other hand, for the fixed-based car simulator, the physical model is implemented within Unreal Engine using the ChaosVehicle framework. This framework provides accurate results and can be configured by a small team with limited experience in dynamic behavior modeling. In summary, the bicycle simulator calculates and applies the physical state externally, while the car simulator relies on Unreal Engine's built-in capabilities. (Epic Games, 2023)

The bicycle simulator receives data from the Unreal Instance about the road gradient and resistance factor of the surface, which affect pedalling resistance and tilting of the bicycle. In the car simulator, the driver's input is processed by the software and sent to the Unreal Instance to simulate the vehicle state. Information such as forward velocity and signal states are fed back to the simulator to modify force feedback and speedometer indication. Both simulators support monitor or HMD visualization.

Replication Methods

To achieve interaction between simulators, data exchange at the Unreal instance level is necessary. A single Unreal project was used for both car simulators, starting at different positions in the level. Data exchange was performed object by object, with each simulator having its own representation in its Unreal instance (ego-perspective) and a corresponding representation in the other simulator's Unreal instance (multi-user perspective).

Figure 2 shows the object replication of the car simulator.

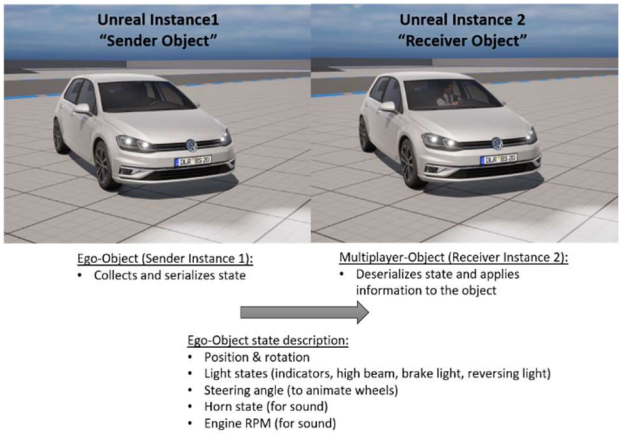

Figure 2 Car Simulator Object Replication

Different approaches were tested for displaying transmitted information in the multiplayer object. Initially, two separate objects were created in Unreal, each with its own functionality. The multiplayer object didn't have physics simulation or control signal processing like the ego object, but it was responsible for displaying additional information (e.g., wheel turning or head movements in the bike simulator). However, this approach required duplicating and maintaining the object, which was time-consuming for small development teams. An improved approach was to create a single implementation that could function as both an ego and multiplayer object, depending on initialization in the virtual environment. This approach involved implementing a server-client concept for information distribution and processing.

Message Broker & Unreal Server

An own developed gRPC-based message broker with unreal integration was used to exchange the information within the Unreal instances. It can dynamically request information from the objects that needs to be replicated. To do this, the Unreal instances register with the broker and inform it about the desired objects. The minimum information that must always be transmitted is the *transform* properties (position/rotation/scaling) of the objects in the virtual world. To replicate additional information of the objects, they are named with an adjustable prefix. The Replicator object recognizes and transmits these to further simulation participants. The Unreal instances involved have authority over their own virtual environment.

When using an unreal dedicated server, the exchange of information is different. The server hosts and has authority over the main world instance and replicates the information of the actors it contains to the connected clients. (The clients don't have the authority to move objects in their own environment. The simulators pass the simulator inputs to the server, which updates all simulator objects (cars, bicycles, pedestrians, etc.). After processing, it updates the client states. The advantage of this procedure is that it can ensure that each client has the same state.

The benefits of the Message Broker are that it is easy to configure, replicate across Unreal projects and enable communication between different objects. Seamless setup, cross-project replication and flexible send and receive capabilities are beneficial in small development teams. When using the dedicated server, Unreal built-in utilities make it easy to use prediction and correction methods to improve networked applications. This can enable multi-user simulation with simulators far apart.

In summary, the different approaches are better suited for different use cases. The message broker approach promises a faster turnaround in smaller Multi-User-Simulations running on a local network. The Unreal Server approach offers better scalability as the number of participants grows, as well as connectivity with slower connection (e.g. the internet).

Conclusion

More experience needs to be gained and changes in technologies taken into account in the development of Multi-User-Simulation. By then, it should be possible to evaluate different approaches to linking simulators and their impact on developing those.

References

Chance, G. et al., 2022. On Determinism of Game Engines Used for Simulation-Based Autonomous Vehicle Verification. *IEEE Transactions on Intelligent Transportation Systems,* Issue DOI: 10.1109/TITS.2022.3177887, pp. 20538-20552.

Epic Games, 2023. *docs.unrealengine.com.* [Online] Available at: https://docs.unrealengine.com/5.1/en-US/vehicles-in-unreal-engine/
[Accessed 11 04 2023].

Muehlbacher, D., 2015. *Multi-Driver Simulation – the link between driving simulation and traffic.* Munich: Conference: mobil.TUM 2015.